3Ж

УДК 004.934.1'1

А.В. Жук, С.С. Панфилов
Институт проблем искусственного интеллекта МОН Украины и НАН Украины, г. Донецк
juk@iai.dn.ua, panfilovss@yandex.ru

Разработка элементов расширенной архитектуры компьютерной системы работы со звуком

В статье сформулированы требования, выдвигаемые в современных условиях к компьютерной системе для работы со звуком, приведена структурная схема такой системы, рассмотрена организация хранения звуковых данных. В качестве промежуточного результата разработки архитектуры модуля обработчика приведена объектная модель цепочки обработчиков звуковых данных.

Введение

На сегодняшний день разрабатывается множество проектов, связанных с цифровой обработкой звуковых сигналов и основанных на ней задачах: идентификацией диктора, распознаванием речи, музыки и т.д. Существует ряд библиотек, которые инкапсулируют сам процесс обработки [1-3], реализуя некоторую внутреннюю программную среду для хранения промежуточных результатов и обмена данными. В то же время для исследователя, к примеру, в области распознавания речи, использование подобных библиотек возможно лишь для сравнения качества своих результатов с результатами распознавателя, реализованного в библиотеке. Исследователь создаёт свою библиотеку, свой программный комплекс, в котором, и это совершенно естественно, будут использованы специфические структуры данных, механизмы обмена ими, механизмы оповещения и т.д. Это приводит к ситуации, когда результаты разработок по одной и той же тематике, полученные разными исследователями, оказываются несовместимыми на уровне интерфейсов, и это – в лучшем случае, поскольку несовместимость может находиться выше – на уровне средств разработки. Решение таких ситуаций требует дополнительных временных и материальных затрат, причём зачастую, если отсутствует несовместимость на уровне средств разработки, проблема решается просто путём внедрения части программного кода из одного проекта в другой проект. В случае взаимодействия двух учреждений в рамках одного проекта помимо указанных выше сложностей возникает необходимость в сохранении авторских прав на программную реализацию того или иного алгоритма, что автоматически исключает возможность слияния части программного кода.

Всё вышесказанное определяет актуальность создания такой архитектуры компьютерной системы работы со звуком, которая бы:

1) изначально по своей структуре была модульной;
2) предусматривала относительно простое внедрение новых модулей от сторонних разработчиков;
3) подразумевала сохранение авторских прав разработчиков модулей;
4) по возможности устраняла проблемы с несовместимостью форматов данных, используемых в обмене информацией между модулями;
5) обеспечивала обратную совместимость форматов данных и интерфейсов модулей в различных версиях протокола данной архитектуры;
6) предоставляла возможность избирательного оповещения модулей об изменении обрабатываемых данных;
7) позволяла формировать произвольные последовательности обработки с использованием имеющихся в наличии модулей.

Разработка архитектуры компьютерной системы предполагает создание спецификаций на форматы данных, протоколы обмена ими, интерфейсы модулей и т.д.

Целью данной работы является проектирование общей структуры компьютерной системы работы со звуком, а также определение формата хранения звуковых данных и разработка объектной модели обработчиков звука в рамках вышеупомянутой архитектуры.

Проектирование общей структуры системы

Как было сказано выше, одним из требований, предъявляемых к компьютерной системе работы со звуком, является её модульная структура. С точки зрения проектируемой архитектуры модулями являются абсолютно все части обычной системы цифровой обработки звука: отображение, загрузка и сохранение, воспроизведение и запись, обработка и распознавание и т.д. Эти модули объединяются вокруг ядра системы, которое реализует функции хранилища данных. В хранилище могут находиться данные как предопределённых, так и пользовательских типов. Предопределённые типы данных предлагается сделать расширяемыми. Причём базовые характеристики расширенного типа данных должны быть доступными модулю, предназначенному для обработки базового типа данных. Такую расширяемость достаточно легко реализовать через полиморфизм при наследовании [4]. В качестве предопределённых типов данных для предметной области, связанной с цифровой обработкой звуковых сигналов, выступают, как минимум:

1) звуковой сигнал;

2) метки, отражающие внутреннюю структуру сигнала.

Основным механизмом взаимодействия между модулями должны быть сообщения, причём каждый модуль должен иметь возможность определять список сообщений, которые он должен получать.

В состав системы должны также входить диспетчер модулей и диспетчер очерёдности обработки. Диспетчер модулей – это подсистема, предназначенная для организации загрузки модулей в процессе работы системы, а также для перенаправления сообщений. Диспетчер очерёдности обработки – это подсистема, основным назначением которой является формирование произвольных последовательностей обработки с использованием имеющихся в наличии модулей, а также организация выполнения заданной последовательности обработки.

Предлагаемая структура компьютерной системы работы со звуком изображена на рис. 1.

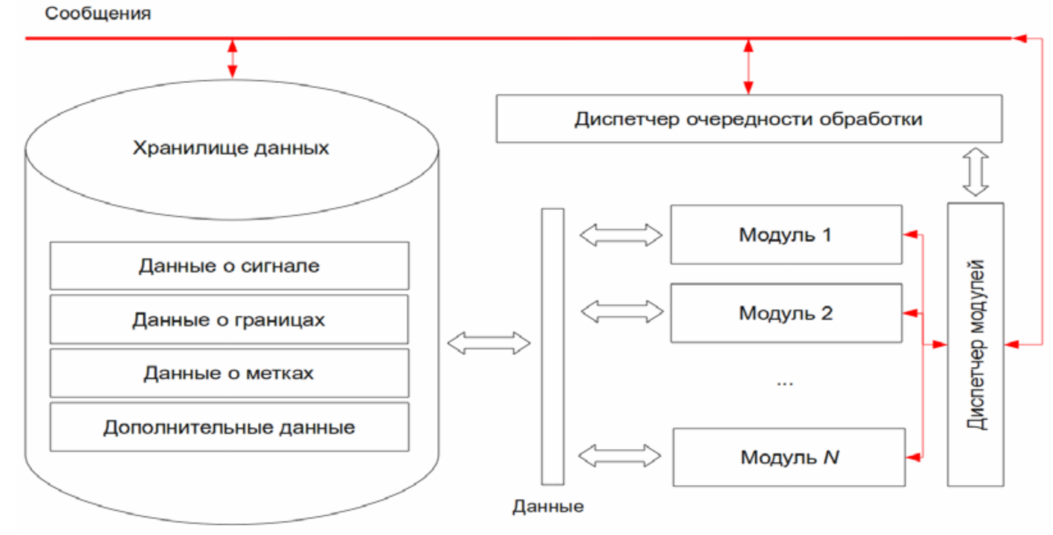


Рисунок 1 – Предлагаемая структура компьютерной системы работы со звуком

Формат хранения звуковых данных

Современные звуковые карты позволяют захватывать и воспроизводить звуковые данные в нескольких различных форматах, получаемых как комбинации допустимых значений частоты дискретизации и глубины квантования. Возможные значения этих параметров приведены в табл. 1.

Таблица 1 – Возможные значения частоты дискретизации и глубины квантования

Частота дискретизации (Гц)	Глубина квантования (бит)
8000	8
11025	16
22050	*24*
44100	
48000	
96000	

Курсивом в табл. 1 обозначены значения параметров, получивших широкое распространение в последнее время, но не поддерживаемых форматом PCM [5]. Безусловно, оба упомянутых показателя играют важную роль при АЦП и ЦАП [6]. В то же время в цифровой обработке звука методы работы с частотным представлением сигнала активно оперируют его частотой дискретизации, а вот глубина квантования в основном влияет на тип элементов массива, представляющего звуковой сигнал в памяти ЭВМ. Необходимость поддержки разных значений глубины квантования приводит к необходимости создания нескольких одинаковых функций, принимающих в качестве параметров массивы элементов разного типа. Более того, при запуске обработчика звука нужно выбирать правильную функцию для текущей глубины квантования. Частично справиться с этим позволяет использование шаблонных функций, что убирает дублирование кода, однако необходимость в выборе правильных шаблонных параметров при вызове таких функций всё равно остаётся. Ещё одним существенным недостатком шаблонных функций является невозможность их использования в динамических библиотеках в качестве экспортируемых.

В качестве альтернативного подхода можно предложить использование для описания одного отсчёта звука типа данных, преобразование в который не вызовет потери информации для любого из используемых на сегодня значений глубины квантования. Следует заметить, что подобный подход применяется и в современных звуковых редакторах. Как показала практика, наибольшую общность в данном случае обеспечивает хранение последовательности отсчётов звука в виде массива чисел с плавающей точкой, принимающих значения из интервала [-1;1]. Поэтому именно такое представление предлагается использовать для хранения звуковых данных в хранилище и при обмене данными между модулями.

В качестве основы для хранилища данных рассматриваемой системы была разработана иерархия классов, изображенная на рис. 2.

Основным здесь является класс SoundContainer. Он обеспечивает хранение звуковых данных (в виде структуры SoundBuffer), а также удобный доступ к ним. Кроме того, данный класс способен принимать и возвращать данные разной глубины квантования. Структура SoundBuffer служит для хранения массива отсчётов звукового сигнала.

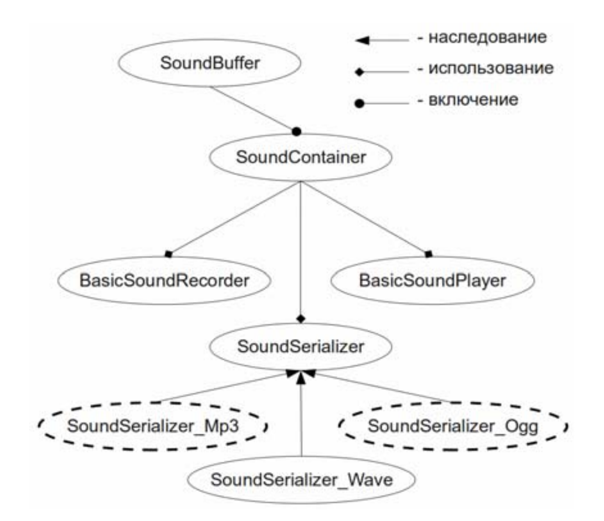


Рисунок 2 – Диаграмма классов, обеспечивающих хранение звуковых данных

Список полей этой структуры приведен в табл. 2. Именно SoundBuffer выступает в качестве основного средства обмена звуковыми данными в рассматриваемой системе работы со звуком.

Таблица 2 – Поля структуры SoundBuffer

№ п/п	Тип данных	Имя переменной	Описание
1.	float*	pData	Массив отсчётов звуковых данных. Все значения принадлежат промежутку [-1;1].
2.	int	iSize	Количество элементов в массиве pData.
3.	int	iSampleRate	Частота дискретизации звукового сигнала, содержащегося в данном буфере.
4.	bool	bDisposable	Флаг, указывающий, нужно ли в деструкторе очищать выделенную под pData память.

SoundSerializer – это базовый класс, обеспечивающий интерфейс для работы с мультимедийными файлами. Основные операции – это сохранение и загрузка. Потомки данного класса должны реализовывать поддержку различных форматов звуковых файлов и методов сжатия звука. Реализация класса SoundSerializer_Wave сделала возможной работу со звуковыми файлами формата PCM Wave. Классы SoundSerializer_Mp3 и SoundSerializer_Ogg приведены как примеры возможных расширений функциональности системы.

Классы BasicSoundPlayer и BasicSoundRecorder представляют собой реализации интерфейсов для записи и воспроизведения звука, в них реализована работа с библиотекой OpenAL [7].

Объектная модель цепочки обработчиков звуковых данных

На данный момент разработана объектная модель последовательности обработчиков звуковых данных, достаточно близко реализующая ту часть приведенной на рис. 1 системы работы со звуком, которая отвечает за подключение модулей и определение последовательности использования модулей при обработке звука (диспетчер модулей и диспетчер очередности обработки). В данной модели класс Chain является как диспетчером очередности обработки, так и диспетчером модулей. Класс Action является базовым классом для любого обработчика звука и определяет интерфейс, который должен предоставлять каждый такой обработчик. При разработке этого интерфейса были исполь-

зованы соображения следующего характера: обработчик звука определяется и оперирует тремя характеристиками: типом обработчика, параметрами обработчика и собственно обрабатываемым звуком. Тип обработчика задаётся классом, его реализующим. Таким образом, обработчику указанного типа необходимо передать параметры обработки и обрабатываемый звуковой сигнал. Описание функций, входящих в класс Action, приведено в табл. 3.

Таблица 3 – Функции класса Action

№ п/п	Тип возвращаемого значения	Имя функции	Принимаемые параметры	Описание
1.	void	SetParams	sActionParams* newParam	Устанавливает множество параметров обработчика. Содержимое sActionParams должно соответствовать типу обработчика. Копирование данных при этом не производится.
2.	void	Apply	SoundBuffer& sbData	Запускает обработку звука, содержащегося в sbData. Результат обработки помещается в sbData.
3.	sActionParams*	GetParams	–	Возвращает множество параметров обработчика. Как и в случае с SetParams, копирование данных не производится

Диаграмма классов [8] этой объектной модели приведена на рис. 3.

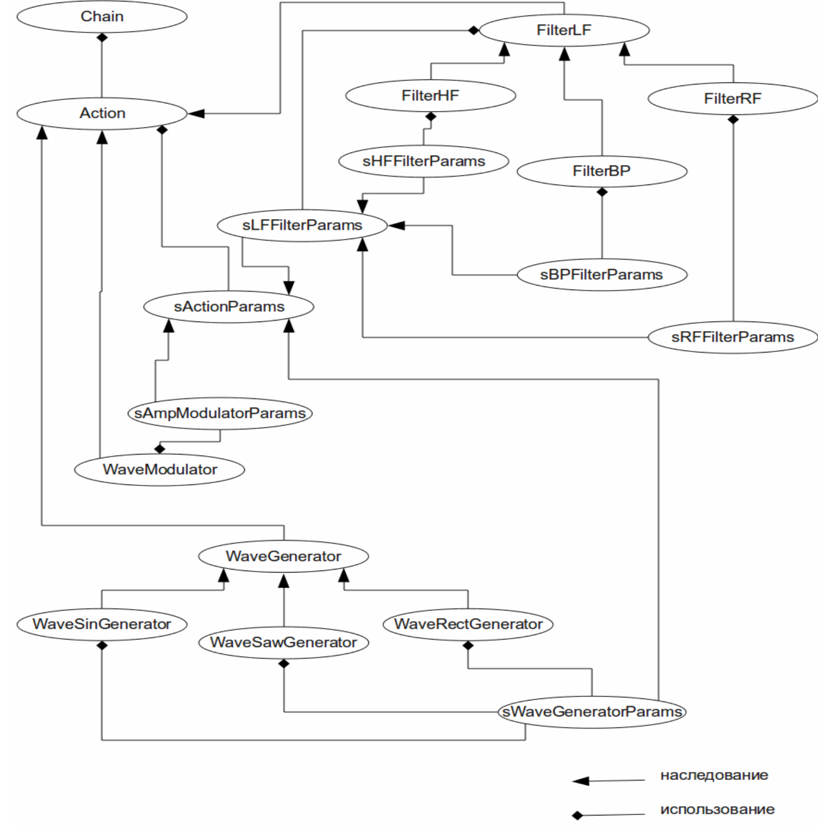


Рисунок 3 – Диаграмма классов объектной модели последовательности обработчиков звуковых данных

Класс Action содержит единственное поле – указатель на структуру sActionParams.

Возможность передачи различным обработчикам разного набора данных в качестве параметров реализована через иерархию структур, базовой в которой является структура sActionParams. Эта структура содержит только одно поле данных – переменную Type. Данная переменная объявлена как защищённая, инициализируется в конструкторе и предназначена для хранения информации о типе параметризуемого объекта – обработчика звука. Такой подход позволяет обработчику проверять подаваемые на вход параметры на правильность с точки зрения используемых типов данных. Недостатком такого подхода на данный момент является невозможность проверить подаваемые данные не на полное соответствие, а на совместимость. Это можно было бы сделать, реализовав сохранение в sActionParams легко получаемой информации об иерархии наследования. В этом случае можно было бы искать первого общего предка, изымать из полученных данных общую на уровне найденного предка информацию, а все остальные, непроинициализированные, значения параметров устанавливать по умолчанию.

Выводы

В работе были рассмотрены требования, выдвигаемые к архитектуре компьютерной системы работы со звуком, спроектирована структура такой системы, описаны объектные модели хранилища и последовательности обработчиков звуковых данных. Недостатком полученных в результате реализации этих моделей частей программного кода является необходимость их статического включения в проект (хотя даже на этом этапе стыковка фрагментов кода, написанных разными разработчиками, значительно упростилась). Дальнейшие работы предполагают разработку спецификации сообщений системы, уточнение модели хранилища данных и спецификаций интерфейсов модулей, добавление возможности идентификации типов используемых данных во время выполнения.

Литература

1. LADSPA [Электронный ресурс]. – Режим доступа : http://ru.wikipedia.org/wiki/LADSPA
2. Virtual Studio Technology [Электронный ресурс]. – Режим доступа : URL: http://ru.wikipedia.org/wiki/Virtual_Studio_Technology
3. DirectX [Электронный ресурс]. – Режим доступа : http://ru.wikipedia.org/wiki/DirectX
4. Пол А. Объектно-ориентированное программирование на С++/ Пол А. – СПб. : Бином, 2001. – 464 с.
5. Радзишевский А.Ю. Основы аналогового и цифрового звука / Радзишевский А.Ю. – М. : Издательский дом «Вильямс», 2006. – 288 с., ил.
6. Рабинер Л.Р. Цифровая обработка речевых сигналов / Л.Р. Рабинер, Р.В. Шафер; [пер. с англ. под ред. М.В. Назарова и Ю.Н. Прохорова]. – М. : Радио и связь, 1981. – 496 с., ил.
7. OpenAL [Электронный ресурс]. – Режим доступа : URL : http://ru.wikipedia.org/wiki/OpenAL
8. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на С++/ Буч Г. – СПб. : Бином, 1998. – 560 с.

О.В. Жук, С.С. Панфілов

Розробка елементів розширеної архітектури комп’ютерної системи роботи зі звуком

В статті сформульовано вимоги, які висуваються до комп’ютерної системи роботи зі звуком в сучасних умовах, наведено структурну схему такої системи, розглянуто організацію збереження звукових даних. Який проміжний результат створення архітектури модуля-обробника наведено об’єктну модель послідовності обробників звукових даних.

A. Zhuk, S. Panfilov

Development of Extended Computer System Architecture for Sound Processing

There the requirements to modern computer system for sound processing are formulated in the article, the structure of such a system and sound data storage organization are discussed. Also, object-oriented model of sound data processors sequence is showed as an intermediate milestone towards sound processor module architecture.

Статья поступила в редакцию 08.07.2010.